Рецензия
на материалы авторской методической разработки
«Методические рекомендации по информатике по анализу программы, содержащей подпрограммы, циклы и ветвления»
учителя информатики МБОУ лицей №4
им. профессора Е.А. Котенко г. Ейска МО Ейский район
Вороновой Ирины Николаевны

Материалы методического пособия предназначены для учителей информатики, работающих в 9–11 классах, при подготовке к сдаче ЕГЭ в новой компьютерной форме. Они также будут полезны обучающимся 10–11 классов, планирующих сдавать ЕГЭ по предмету.

Главная задача учителя информатики — организовать работу с обучающимися таким образом, чтобы выбор предмета «информатика» на государственной итоговой аттестации был осознанным и правильным, создать условия для обеспечения качественной подготовки обучающихся и успешной сдачи ими ЕГЭ по информатике и ИКТ.

Количество страниц — 43.

Цель методических рекомендаций — помочь учителю в подготовке обучающихся, планирующих сдачу ЕГЭ и изучающих язык программирования Паскаль, к решению задач, требующих знаний по программированию, умения составлять и анализировать компьютерные программы.

Актуальность и востребованность материалов методического пособия состоит в том, что сейчас всё более популярным становится язык программирования Python, всё больше материалов для подготовки к ЕГЭ появляется именно на этом языке, позволяющем решать задачи, написав малое количество кода. Казалось бы, что надо просто использовать в обучении информатике именно этот язык. Однако Python имеет весьма существенные недостатки, и сделать однозначный выбор в его пользу нельзя. Многие школы изучают язык Паскаль. Но в своём классическом варианте он устарел. В рецензируемом методическом пособии решение задач рассматривается на современной версии языка PascalABC.Net, позволяющей использовать многие возможности, доступные Python, и в то же время свободного от многих недостатков последнего. Предлагаемые автором методические рекомендации частично восполняют имеющийся недостаток материалов по подготовке к ЕГЭ.

Новизна и практическая значимость методического пособия состоит в том, что кроме того, что рассматривается современная версия языка программирования, значительное внимание уделяется сокращённой записи алгоритмов с использованием специфических возможностей языка PascalABC.Net. Это позволяет сократить время набора программы на компьютере и быстрее получить решение задачи. В условиях ограниченности времени экзамена это немаловажно.

Методические рекомендации содержат не только разбор решения 20 задач, охватывающих задания 16 и 23 ЕГЭ, но и задачи для самостоятельного решения с ответами. Это служит более качественному усвоению материала. Разбор типовых задач, примеры их решений, самостоятельные работы с ответами всё направлено не только на преодоление трудностей в изучении темы, но и на организацию качественной подготовки к урокам и к итоговой аттестации.

Рецензируемые материалы методического пособия «Методические рекомендации по информатике по анализу программы, содержащей подпрограммы, циклы и ветвления» Вороновой И.Н. актуальны и могут быть рекомендованы к использованию в работе общеобразовательных организаций Ейского района.

Рецензент
руководитель РМО
учителей информатики — А.Н. Середа

Подпись удостоверяю

Директор — Г.П. Гришко

МКУ «Информационно-
методический центр
системы образования Ейского района»

25.08.2023 г.

Муниципальное бюджетное общеобразовательное учреждение лицей № 4 имени профессора Евгения Александровича Котенко города Ейска муниципального образования Ейский район

Методические рекомендации по информатике по анализу программы, содержащей подпрограммы, циклы и ветвления.

Автор - составитель:
Воронова Ирина Николаевна,
учитель информатики
МБОУ лицей №4 им.
профессора
Е.А. Котенко г.Ейска

2023-2024 учебный год

Оглавление

Предисловие

Данные методические рекомендации предназначены для учителей 10-11 классов, работающих в школе по УМК Босовой Л.Л. базового уровня. Учебники разработаны в соответствии с требованиями ФГОС СОО, в которых изучается программирование алгоритмов на языке Pascal. Несомненно, язык структурного программирования Pascal является одним из лучших языков для начального программирования, но классический язык существенно устарел, в нем отсутствуют модули и возможности современных приложений. В своих рекомендациях я разберу решения задач с использованием PascalABC.NET, современной версии языка, обладающей набором стандартных функций и процедур, краткая запись программ в которой значительно сокращает время решения задач. Использование материалов методических рекомендаций в 11 классе при изучении главы №2 «Алгоритмы и элементы программирования» способствует развитию логического мышления, анализу основных алгоритмических конструкций, содержащих циклы, ветвления и подпрограммы. Данные методические рекомендации могут использоваться на дополнительных занятиях и уроках для подготовки учащихся к сдаче ЕГЭ в новой компьютерной форме.

Пояснительная записка

Перед выпускником общеобразовательной средней школы остро встает вопрос выбора будущей профессии и как следствие этого выбор предметов для сдачи ЕГЭ. Сегодняшние реалии таковы, что актуальным и востребованным направлением являются технологии IT. Подготовка к заданиям ЕГЭ по информатике всегда динамична и требует больших ресурсов, фундаментальных знаний по моделированию, основам логики, теории систем счислений и, конечно же, умению программировать. Встает вопрос, на каком языке программирования лучше готовиться к КЕГЭ.

Популярным на сегодня среди молодёжи является язык Python, который считается одним из самых комфортных для новичков, простой синтаксис и универсальность привлекает внимание широкого круга людей. Однако, жесткая структуризация отступов, великое множество библиотек, отсутствие многомерных массивов, скриптовый интерпретированный язык, динамическая типизация не такие уж безобидные недостатки Python. В первую очередь перед будущим специалистом в области технологий, стоит задача научиться строить алгоритм конкретной задачи, а язык лишь средство реализации этого алгоритма.

Изменения в заданиях КЕГЭ в 2023 году касаются анализа алгоритмов для исполнителя, определения возможного результата работы вычислительных алгоритмов и алгоритмов управления. Бытует мнение, что раз задания выполняются на компьютере, то их следует решать программным кодом. Это, конечно, заблуждение. Многие задания быстрее и легче решать аналитически, используя программные средства, такие как калькулятор в различных режимах, электронные таблицы, текстовый процессор, блокнот, Кумир, графический редактор. Главное дать быстрый и правильный ответ.

Целью методических рекомендаций являются методы анализа программ, содержащих подпрограммы, циклы и ветвления в среде программирования PascalABC.NET.

Актуальность данных методических рекомендаций заключается в том, что большинство заданий КЕГЭ решается с помощью программного кода, содержащего подпрограммы, циклы и ветвления. Учащийся должен показать умение анализировать сложные алгоритмические структуры, составлять программы на выбранном языке программирования. Практически все разборы задач вариантов осуществляются в среде программирования Python, и далеко не всем легко перестроиться на новый синтаксис языка, т.к. школьный курс базового уровня опирается на Pascal. Решение задач вариантов КЕГЭ в среде PascalABC.NET затруднительно найти на просторах Интернета.

Мотивированным ученикам данные рекомендации помогут овладеть современными, более удобными методами решения, которые позволяют быстро и рационально решать многие задания КЕГЭ по информатике.

Практическая значимость методических рекомендаций состоит в помощи учителям и учащимся при изучении основных алгоритмических конструкций, содержащих циклы, ветвления и подпрограммы в среде программирования PascalABC.NET.

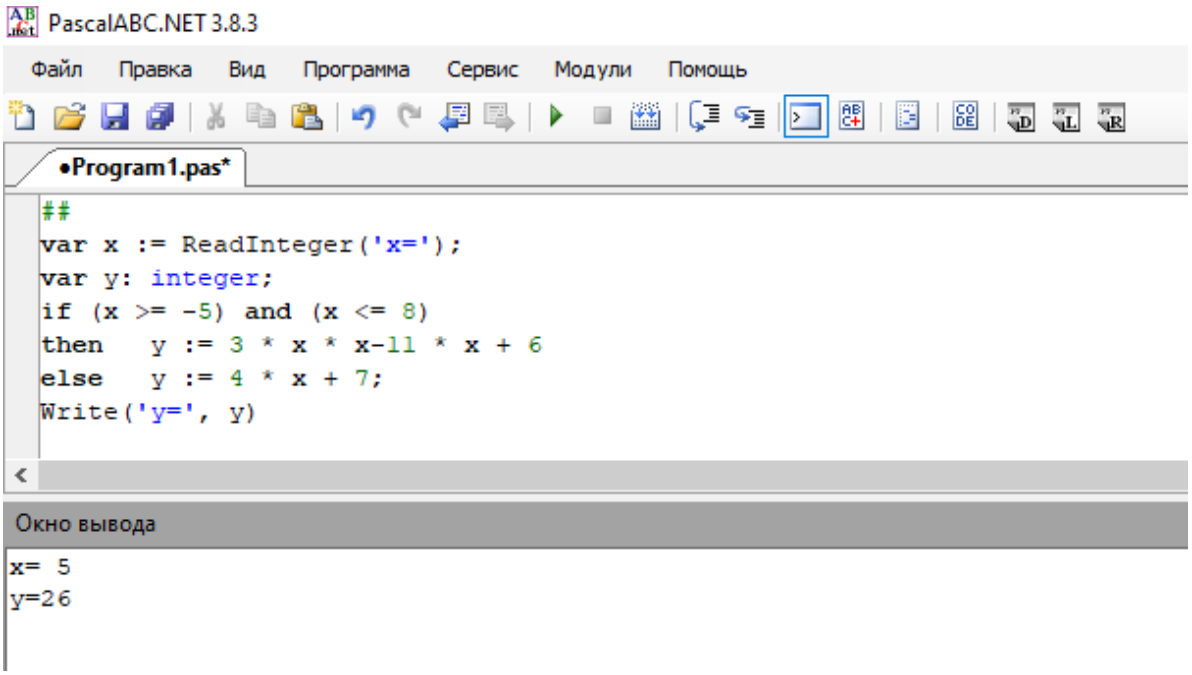
Таким образом основными задачами методических рекомендаций являются:

1. Расширить теоретические знания обучающихся в области основных алгоритмических структур, реализованных в среде программирования PascalABC.NET;
2. Показать возможные методы решения задач с использованием основных алгоритмических структур в среде программирования PascalABC.NET;
3. Сформировать навыки использования циклов, ветвления и подпрограмм в решении конкретных задач в среде программирования PascalABC.NET.

Анализ программы, содержащей подпрограммы, циклы и ветвления.

С помощью основных алгоритмических конструкций можно создавать модели, описывающие реальный мир.

Условный оператор реализует выбор действий, исходя из анализа некоторого условия. Существует два вида формата условного оператора: в одном действия выполняются как по одной, так и по другой ветке; неполное ветвление предполагает действия только при выполнении некоторого условия. В PascalABC.NET существует условная операция, которая по форме очень схожа с условным оператором, однако имеет более компактную форму записи. Например, программа, вычисляющая значение функции через условный



оператор:

Рис.1 Условный оператор в среде программирования PascalABC.NET

Условная операция существенно сокращает запись:

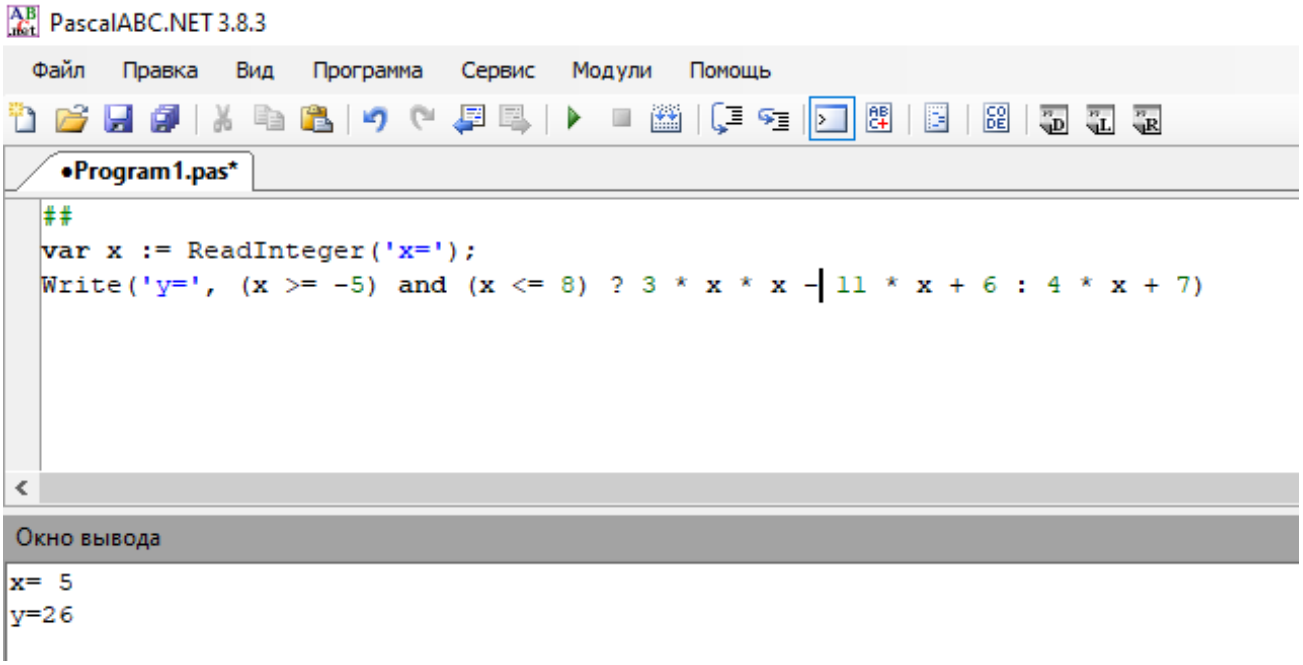


Рис.2 Условная операция в среде программирования PascalABC.NET

Для реализации многих алгоритмов требуется выполнение некоторого числа повторов одних и тех же действий, такие участки программ называют циклом. Язык PascalABC.NET предлагает пять разновидностей операторов цикла:

- Цикл с заданным числом повторений loop // loop <Целочисленное Выражение> do <ТелоЦикла>;
- Цикл с параметром или со счетчиком For имеет шаг +1 или -1// если параметр цикла возрастает, то применяют шаг +1//

  for var <ИмяПараметра> := <Выражение1> to <Выражение2> do begin <ТелоЦикла> end;

  если <Выражение1> больше <Выражение2>, т.е. параметр цикла убывает, применяют шаг -1 //

  for var <ИмяПараметра> := <Выражение1> downto <Выражение2> do begin <ТелоЦикла> end;

- Цикл с предусловием While // while <ЛогическоеВыражение> do <ТелоЦикла>;
- Цикл с постусловием Repeat //repeat <ТелоЦикла> until <ЛогическоеВыражение>;
- Цикл foreach позволяет выполнять переборы всех элементов массива без выяснения размеров массива;

Процедурное программирование предполагает разбиение программы на связанные между собой определенным образом, посредством обращений друг к другу, подпрограммы. В PascalABC.NET можно обращаться к уже имеющимся подпрограммам модулям или создавать пользовательские подпрограммы.

В своих рекомендациях хочу поделиться опытом решения рекурсивных алгоритмов вычислений с помощью компилируемого языка со статической типизацией PascalABC.NET.

Прежде всего хочется предостеречь от решения, составлением программного кода без предварительного анализа функции, чаще всего вызывающие зацикливание или аварийное завершение программы. Если значение функции определяется с возрастающим аргументом с операцией сложения или произведения, возможны «подводные камни», такие как переполнения стека, когда в стеке вызовов храниться больше информации, чем он может вместить. Классическая рекурсивная функция программируется очень просто, по условию задания. Разберу на примерах заданий КЕГЭ №16 и 23.

Методические рекомендации к заданию №16 КЕГЭ

В задании №16 КЕГЭ указывается рекурсивный алгоритм вычисления некоторой функции, обычно целочисленной, предлагается найти значение функции для некоторого аргумента, либо указать количество значений функции, удовлетворяющих определенному значению, возможно на некотором интервале, сумму или произведение значений функции на некотором отрезке. В некоторых заданиях функция может принимать вещественные значения.

Пример 1.

Алгоритм вычисления функции F(n), где n – натуральное число, задан следующими соотношениями:

F(1) = 1;

F(n) = n+F(n / 2), если n чётно;

F(n) = n *F(n – 1), когда n >1 и нечётно.

Чему равно значение F(37)?

Решение предполагает использование втруктуры ветвления в рекурсивном алгоритме:

```
##
function f(n:integer):integer := n = 1 ? 1 :
    n mod 2 = 0 ? n + f(n div 2) :
    n * f(n - 1);
print(f(37));
```

Ответ: 6993.

Пример2.

Алгоритм вычисления функции F(n), где n – натуральное число, задан следующими соотношениями:

F(1) = 1;

F(n) = F(n – 1)+ n *F(n – 1), когда n >1.

Чему равно значение F(5997)/ F(5995)?.

Решение:

Задача требует анализа функции, иначе в ожидании ответа можно потерять значительное время.

##

```
function f(n:integer):integer := n = 1 ? 1 : n > 1 ? f(n-1) + n * f(n - 1) : 0;
println(f(5997) / f(5995));
```

Здесь аналитическое решение очевидно:

f(5997)= f(5996)+ 5997* f(5996);

f(5996)= f(5995)+ 5996* f(5995);

тогда f(5997)/ f(5995)=

(f(5995)+ 5996* f(5995)+ 5997* (f(5995)+ 5996* f(5995)))/ f(5995);

Вынесем в числителе за скобку f (5995), а затем сократим и получим

f(5995)*(5997+5997*5997)/ f(5995)=5997+5997*5997= 35 970 006

Ответ: 35 970 006.

Пример 3.

Алгоритм вычисления функций F(n) и G(n) задан следующими соотношениями:

F(n) = n, при n <50;

F(n) = 2· G(50-n//2), если n > 49;

G(n) = 10 при n >40;

G(n) = 30+F(n+600// n), если n <41

Чему равно значение F(80)?

Решение:

Рассмотрим рекурсию от двух функций.

```
##
function f(n:integer):integer;
forward;
function g(n:integer):integer;
forward;
function f(n:integer):integer := n <50 ? n : n > 49 ? 2*g(50-n div 2 ) : 0;
function g(n:integer):integer := n >40 ? 10 : n <41 ? 30+ f(n+600 div n) : 0;
print(f(80));
```

Ответ: 812.

Пример 4.

Алгоритм вычисления функции F(n), где n – неотрицательное число, задан следующими соотношениями:

F(n) = n – 1, при n ≤ 3;

F(n) = F(n – 2) + n / 2 – F(n – 4), если n > 3 и n чётно;

F(n) = F(n – 1) · n + F(n – 2), если n > 3 и n нечётно.

Чему равно значение выражения F(4952) + 2 · F(4958) + F(4964)?

Решение:

Программное решение здесь осложнено большими вычислениями, код «висит», однако аналитическое решение очевидно:

f(4952)+ 2*f(4958)+ f(4964);

Определим f(4964)= f(4962)+4964/2- f(4960);

f(4962)= f(4960)+4962/2- f(4958);

Тогда, f(4964)= f(4960)+4962/2- f(4958)+4964/2- f(4960);

Подставим в исходное выражение:

f(4952)+ 2*f(4958)+ 4962/2- f(4958)+4964/2= f(4952)+ f(4958)+ 4962/2+4964/2;

f(4958)= f(4956)+4958/2- f(4954);

f(4956)= f(4954)+4956/2- f(4952);

f(4958)= f(4954)+4956/2- f(4952)+4958/2- f(4954);

и получим

f(4952)+ f(4958)+ 4962/2+4964/2= f(4952)+ 4956/2- f(4952)+4958/2+ 4962/2+4964/2= 4956/2+ 4958/2+ 4962/2+4964/2=9920.

Ответ: 9920.

Пример 5.

Обозначим частное от деления натурального числа a на натуральное число b как a // b, а остаток как a%b. Алгоритм вычисления функции F(n), где n – натуральное число, задан следующими соотношениями:

F(n) = n, если n < 2;

F(n) = n % 2 + 10· F(n//2), если n ≥ 2.

Определите значение n, для которого функция F(n) равна 100000100001000100101.

Решение:

Можно по условию задачи определить, что алгоритм переводит десятичное число в двоичный код, однако, не для всех учеников это очевидно. Тогда надо прибегнуть к исследованию алгоритма.

Напишем программный код данного алгоритма.

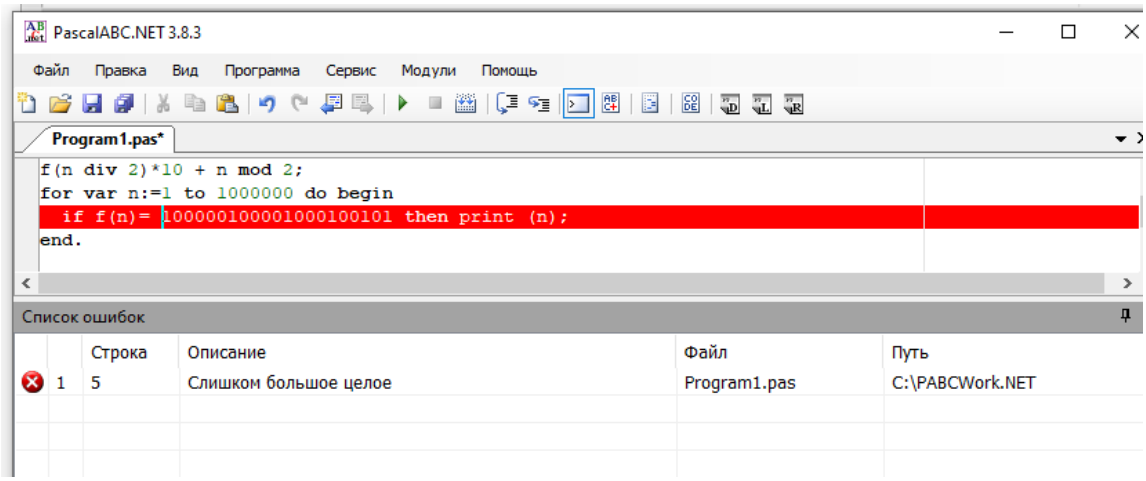


Рис. 3 Программный код решения задачи выдает ошибку.

Проанализируем с различными значениями программу:

##

function f(n:integer):integer := n <2 ? n :

f(n div 2)*10 + n mod 2;

```
print(f(2));
```

ответ: 10

##

```
function f(n:integer):integer := n <2 ? n :
        f(n div 2)*10 + n mod 2;
print(f(20));
```

ответ: 10100

##

```
function f(n:integer):integer := n <2 ? n :
        f(n div 2)*10 + n mod 2;
print(f(128));
```

ответ: 10000000

Понимаем, что программа переводит десятичное число в двоичное, значит можно перевести с помощью того же калькулятора обратно и получить

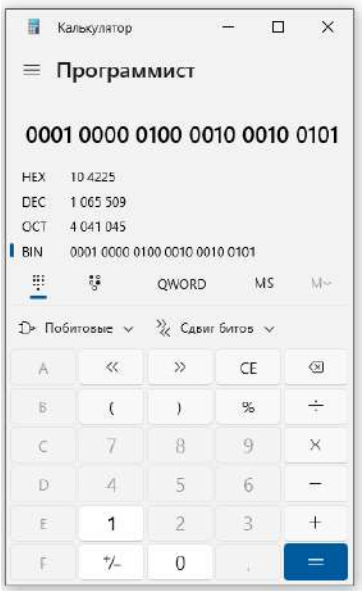


Рис.4 Вычисления на калькуляторе в режиме программист.

Пример 6.

Алгоритм вычисления функции F(n), где n – целое неотрицательное число, задан следующими соотношениями:

F(n) = n при n <10

F(n) = n%10* F(n//10) при n >9

Найдите количество 20-значных чисел n, для которых значение F(n) будет равно 25?

Решение:

Решение в «лоб», написанием программного кода приведет к зависанию, так как работать надо с большими числами.

##

```
function f(n:biginteger):biginteger := n <10 ? n : n>9? f(n div 10)*F(n mod 10): 0;
var c:=0;
for var n:=10000000000000000000 to 10000000000000000000 do
  if f(n)=25 then c+=1;
    println(c);
```

В этом случае без анализа не обойтись. Для, тех учащихся, кто решает задачи программным кодом достаточно много, наверняка понятно, что данная функция определяет произведение цифр числа и очевидно, что результатом будет 25, если в числе две пятерки, а остальные цифры – единицы. Тогда ответом будет число сочетаний двух цифр из 20. 20!/18!*2!=190 Но можно обойтись без формул комбинаторики, произведя анализ работы программы при различных интервальных значениях:

от 10 до 100 результат один;

от 100 до 1000 результатов три;

от 1000 до 10000 результатов шесть;

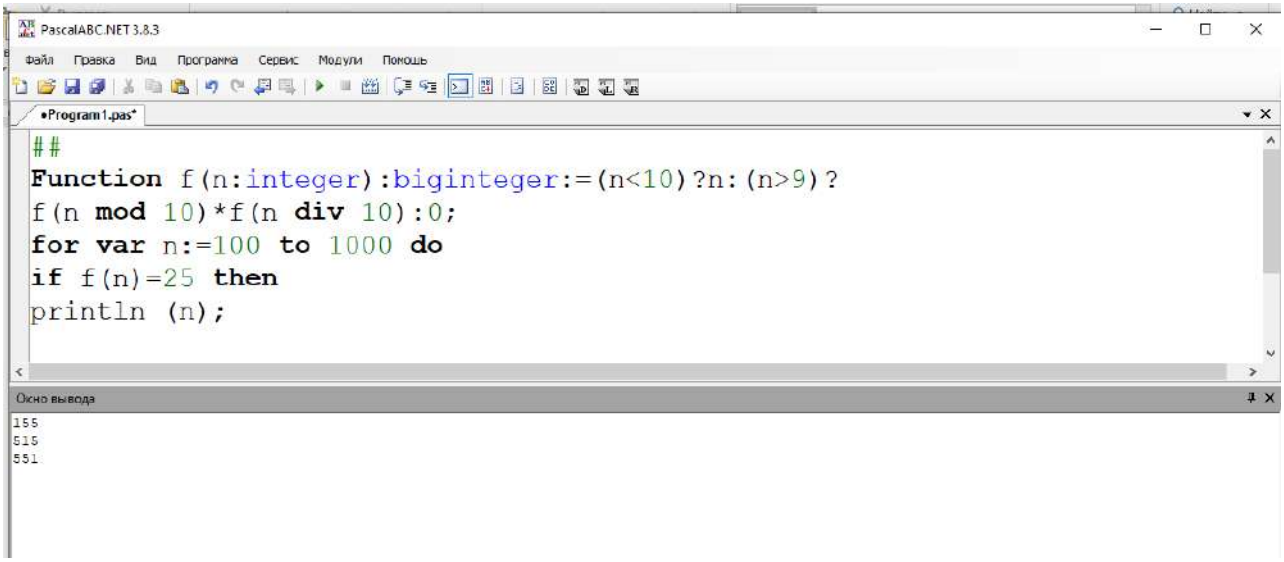
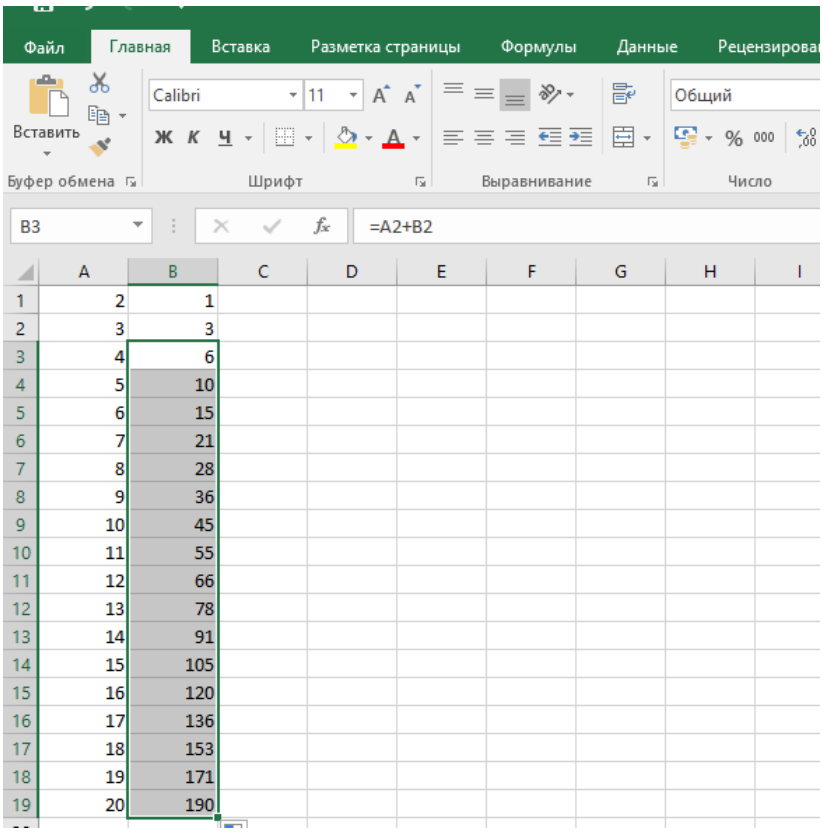


Рис5. Исследование работы программного кода

Воспользуемся, для облегчения вычислений, электронной таблицей и получим количество результатов равных 25 для всех 20-значных чисел, равное 190.



	A	B	C	D	E	F	G	H	I
1	2	1							
2	3	3							
3	4	6							
4	5	10							
5	6	15							
6	7	21							
7	8	28							
8	9	36							
9	10	45							
10	11	55							
11	12	66							
12	13	78							
13	14	91							
14	15	105							
15	16	120							
16	17	136							
17	18	153							
18	19	171							
19	20	190							

Рис 7. Вычисления с использованием электронной таблицы.

Пример 7.

Алгоритм вычисления функции F(n), где n – натуральное число, задан следующими соотношениями:

F(n) = 1, если n ≥ 10000,

F(n) = F(n + 3)+7, если n < 10000 и чётное.

F(n) = F(n + 1)-3, если n < 10000 и нечётное.

Чему равно значение выражения F(50) – F(57)?

Решение:

##

```
function F(n:integer):integer:= n>=10000?1:(n mod 2 = 0) and (n < 10000)?F(n+3)+7:(n mod 2 <> 0) and (n < 10000)?F(n+1)-3:0;
print(F(50)-F(57));
```

Ответ: 11.

Пример 8.

Алгоритм вычисления значения функции F(n), где n – целое неотрицательное число, задан следующими соотношениями:

F(0) = 2

F(n) = F(n–1), при 0 < n ≤ 15

F(n) = 1,6*F(n–3), при 15 < n < 95

F(n) = 3,3*F(n–2), при n ≥ 95

Какая цифра встречается чаще всего в целой части значения функции F(33)?

Решение:

В этой задаче надо обратить внимание, что функция принимает вещественные значения. Написав программный код, выводим результат и подсчитываем чаще всего встречающуюся цифру в целой части результата 33.554432. Ответ очевиден и время на решение затрачено минимальное.

##

function F(n:integer): real := n = 0?2: n <= 15? F(n - 1): n < 95? 1.6 * F(n - 3): 3.3 * F(n - 2);

F(33).Print

Ответ: 3.

Пример 9.

Алгоритм вычисления значения функции F(n), где n – целое число, задан следующими соотношениями:

F(n) = n, при n < 2,

F(n) = F(n/2) + 1, когда n ≥ 2 и чётное,

F(n) = F(3·n + 1) + 1, когда n ≥ 2 и нечётное.

Назовите количество значений n на отрезке [1;100], для которых F(n) определено и больше 100.

Решение:

Исходя из постановки вопроса, можно предположить, что существуют такие значения n, для которых F(n) невозможно вычислить. Наличие операций умножения и сложения в выражении для аргумента n при рекурсивном определении F(n) может привести к проблеме при нечетном n, начиная с 3, потому что тут аргумент в рекурсивном определении функции возрастает. Но как только вычисленный аргумент принимает четное значение, мы на следующем шаге рекурсии попадаем в ветку, где происходит деление, поэтому сказать наверняка, что аргумент постоянно возрастает и этот процесс будет бесконечным, мы не можем.

Программируем функцию.

##

function F(n:integer): integer := n < 2? n: (n mod 2=0) and (n>=2)? F(n div 2) + 1: F(3 * n + 1) + 1;

var c:=0;

for var n:=1 to 100 do

if F(n)>100 then c+=1;

c.print;

Ответ:15.

В этой задаче фраза «для которого F(n) определено» не привела к сложности решения классическим кодом.

Пример 10.

Алгоритм вычисления значения функции F(n), где n – целое число, задан следующими соотношениями:

F(n) = n, при n ≤ 5,

F(n) = n + F(n/5 + 1), когда n > 5 и делится на 5,

F(n) = n + F(n + 6) , когда n > 5 и не делится на 5.

Назовите минимальное значение n, для которого F(n) определено и больше 1000.

Решение:

Пробуем написать программный код.

```
##
function F(n:integer): integer :=n <= 5?n: n mod 5 = 0? n + F(n div 5 + 1): n + F(n + 6);
for var n:=1 to 1000 do
    if  F(n)>1000 then n.print;
```

Из-за переполнения программного стека получаем сообщение об ошибке.

Исследуем программный код. Программа выдает ошибку при значениях n кратных 6 и (n*6+1). Тогда добавим условие:

```
##
function F(n:integer): integer :=(n <= 5) and (n mod 6 > 1)?n: (n mod 5 = 0) and (n mod 6 > 1)? n + F(n div 5 + 1): n mod 6 > 1?n + F(n + 6):0;
for var n:=1 to 1000 do
if  F(n)>1000 then n.print;
```

Ответ 131.

Задания для тренировки к №16

1. Алгоритм вычисления функций F(n) и G(n) задан следующими соотношениями:

$F(n) = G(n) = 1$ при $n = 1$

$F(n) = F(n–1) – 2 \cdot G(n–1)$, при $n > 1$

$G(n) = F(n–1) + G(n–1) + n$, при $n > 1$

Чему равна сумма цифр значения функции G(36)?

2. Алгоритм вычисления функций F(n) и G(n) задан следующими соотношениями:

$F(n) = G(n) = 1$ при $n = 1$

$F(n) = F(n–1) + 3 \cdot G(n–1)$, при $n > 1$

$G(n) = F(n–1) – 2 \cdot G(n–1)$, при $n > 1$

Чему равна сумма цифр значения функции F(18)?

3. Алгоритм вычисления функции F(n) задан следующими соотношениями:

$F(n) = n$ при $n \leq 3$;

$F(n) = n // 4 + F(n–3)$ при $3 < n \leq 32$;

$F(n) = 2 \cdot F(n–5)$ при $n > 32$

Здесь // обозначает деление нацело. В качестве ответа на задание выведите значение F(100).

4. Алгоритм вычисления функции F(n) задан следующими соотношениями:

$F(n) = n$ при $n \leq 3$;

F(n) = n * n * n + F(n – 1), если n > 3 и дает остаток 0 при делении на 3

F(n) = 4 + F(n // 3), если n > 3 и дает остаток 1 при делении на 3

F(n) = n * n + F(n – 2), если n > 3 и дает остаток 2 при делении на 3

Здесь // обозначает деление нацело. В качестве ответа на задание выведите значение F(100).

5. Алгоритм вычисления функции F(n) задан следующими соотношениями:

F(n) = n при n ≤ 10;

F(n) = n // 4 + F(n–10) при 10 < n ≤ 36;

F(n) = 2 · F(n–5) при n > 36

Здесь // обозначает деление нацело. В качестве ответа на задание выведите значение F(100).

6. Алгоритм вычисления функции F(n) задан следующими соотношениями:

F(n) = n при n ≤ 3;

F(n) = 2 · n + F(n – 1) при чётных n > 3;

F(n) = n · n + F(n – 2) при нечётных n > 3;

Определите количество натуральных значений n из отрезка [1; 100], при которых значение F(n) кратно 3.

7. Алгоритм вычисления функции F(n) задан следующими соотношениями:

F(n) = n при n ≤ 3;

F(n) = n + 3 + F(n – 1) при чётных n > 3;

F(n) = n · n + F(n – 2) при нечётных n > 3;

Определите количество натуральных значений n из отрезка [1; 1000], при которых значение F(n) кратно 7.

8. Алгоритм вычисления функции F(n) задан следующими соотношениями:

F(n) = 1 при n ≤ 1;

F(n) = n · F(n – 1) при чётных n > 1;

F(n) = n + F(n – 2) при нечётных n > 1;

Определите значение F(84).

9. Алгоритм вычисления функции F(n) задан следующими соотношениями:

F(n) = 1 при n ≤ 1;

F(n) = n + F(n – 1) при чётных n > 1;

F(n) = n · n + F (n – 2) при нечётных n > 1;

Определите значение F(80).

10. Алгоритм вычисления функции F(n) задан следующими соотношениями:

F(n) = n · n – 5 при n > 15

F(n) = n · F(n+2) + n + F(n+3), если n <= 15

Чему равна сумма цифр значения функции F (1)?

Ответы к заданиям

1. 40
2. 46
3. 655360
4. 121757
5. 180224
6. 32
7. 285
8. 148176
9. 85400
10. 48

Методические рекомендации к заданию №23 КЕГЭ

Обычно после разбора и тренировки заданий №16 КЕГЭ перехожу на разбор заданий №23, которые основаны на работе некоторых исполнителей, имеющих две или более команды. В заданиях №23 проверяется умение анализировать результат исполнения алгоритма. С помощью динамического программирования в задачах этого типа предлагается определить количество вариантов, либо оптимальный маршрут, при этом не требуется полный перебор вариантов, поскольку запоминаются решения всех задач с меньшими значениями параметров. Последние задания построены на достаточно больших значениях и аналитически бывает достаточно проблематично ответить на поставленный в задаче вопрос, поэтому целесообразно в этих заданиях использовать программный код, самый короткий способ решения – переборный, с помощью все той же рекурсивной функции, но от двух переменных.

Типичная формулировка задачи: исполнитель Калькулятор преобразует некоторое число Х в число У с помощью нескольких команд, допустим двух.

Рассмотрим случай, когда Х<У. При переходе от некоторой точки Х к конечной точке У возможны три варианта:

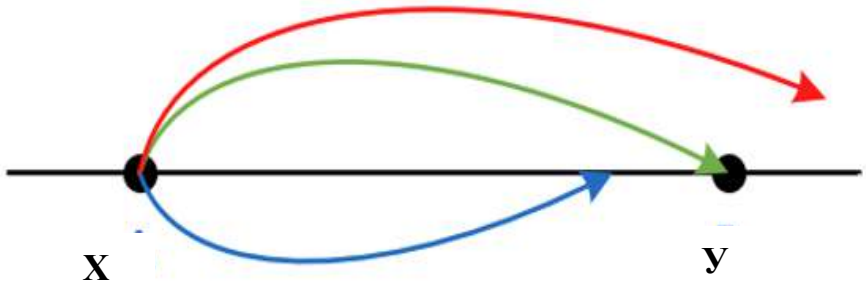


Рис. 8. Варианты попадания начального значения функции.

Можно попасть в интервал от Х до У, можно попасть в значение превышающее У или попасть точно в точку У. таким образом, для рекурсивной функции двух переменных надо рассмотреть интервалы Х <У, Х=У, Х> У. Если

мы попали в У, возвращаем единицу, если перешли У, возвращаем ноль, в интервале от Х до У, возвращаем сумму рекурсивных вызовов функции, заменяя аргумент Х в каждом вызове результатом выполнения каждой из команд исполнителя.

Пример 1.

Исполнитель Июнь15 преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1

2. Умножить на 2

Первая команда увеличивает число на экране на 1, вторая умножает его на 2. Программа для исполнителя Июнь15 – это последовательность команд. Сколько существует программ, для которых при исходном числе 2 результатом является число 29 и при этом траектория вычислений содержит число 14 и не содержит числа 25?

У нас в задании две особые точки – числа 14 (через которое должна проходить траектория) и 25 (а сюда она попасть НЕ должна). Построим дерево решений, расписав подробно ветку умножения. Зачеркнем те ветки, которые не содержат число 14 или, наоборот, попадают в точку 25 или не дают нам конечного значения 29.

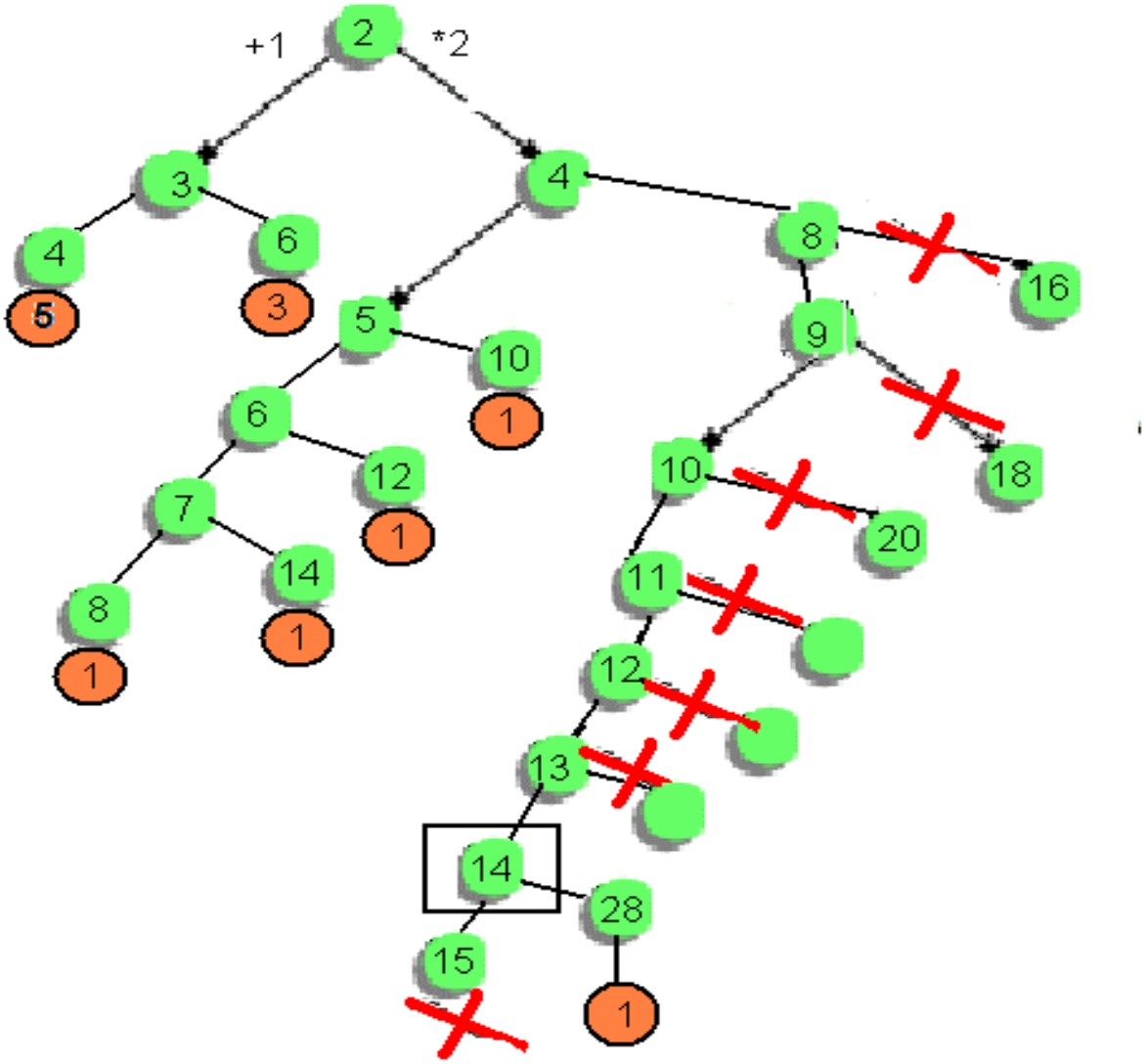


Рис.9. Дерево решений

Введем Функцию (целого типа) подсчета количества цепочек команд (программ), где x – начальное число целого типа, y – конечное число целого типа.

function f(x,y:integer):integer;

Теперь опишем нашу функцию при любых возможных значениях аргументов. Если x=y, т.е. мы находимся в конечном числе (29), то принимаем количество программ равным 1.

Иначе проверяем другое условие. Заметим, что мы идем от меньшего числа к большему, поэтому если x>y, таких программ не существует, как не существует программ, проходящих через точку 25 (по условию задачи) т.е. при x= 25 принимаем количество программ равным 0.

Иначе опишем нашу функцию при x<y, используя две ветки, две команды (+1) и (*2).

Таким образом, мы описали нашу функцию, напишем программу подсчета количества вариантов, проходящих через точку x=14

.

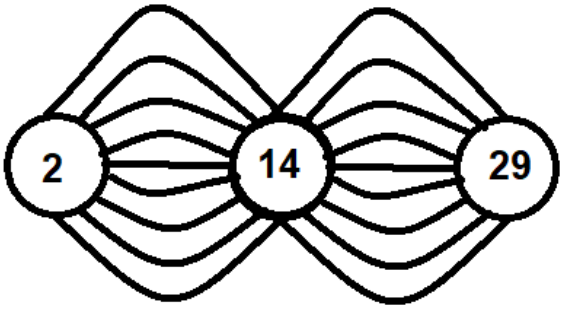


Рис.10. Схема подсчета количества вариантов, проходящих через точку.

```
begin
  print(f(2,14)*f(14,29));
end.
```

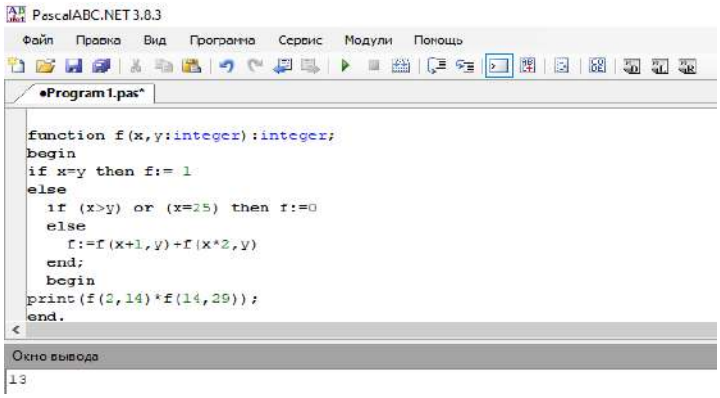


Рис.11. Программный код решения

Или можно записать короче, что сокращает время выполнения задания:

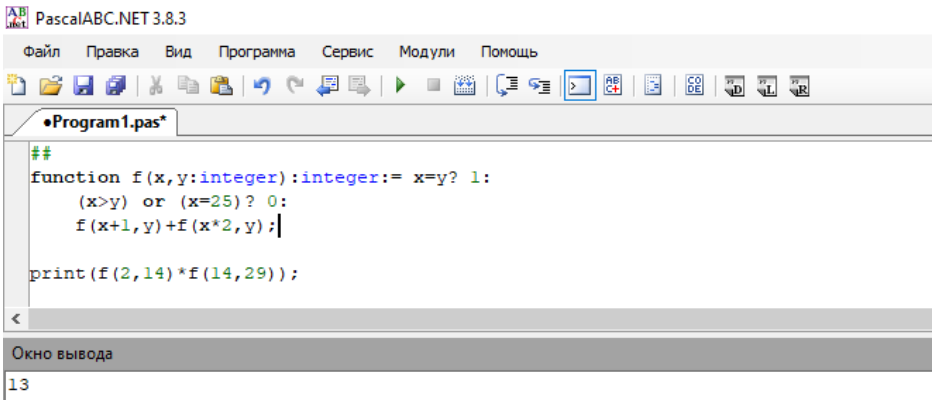


Рис. 12. Краткая запись решения.

Пример 2

Исполнитель Нолик преобразует двоичное число, записанное на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Добавить справа 0
3. Добавить справа 1

Первая команда увеличивает число на 1. При выполнении второй команды, исполнитель справа к числу приписывает 0, а при выполнении третьей команды справа к числу приписывает 1. (например, для числа 10 результатом работы команд №2 и №3 будут являться числа 100 и 101 соответственно). Сколько существует программ, которые исходное двоичное число 100 преобразуют в двоичное число 11101?

Для двоичного числа 100 результатом работы данных команд будут являться числа:

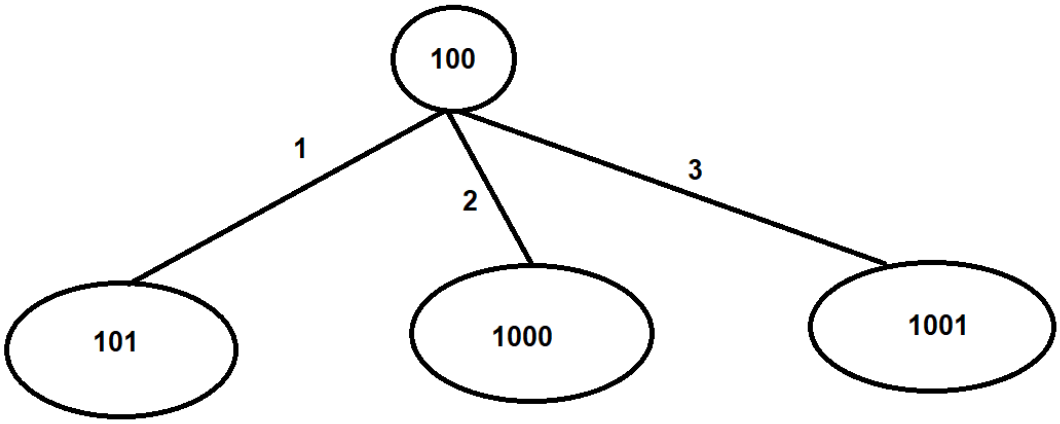


Рис. 13. Варианты результатов команд для двоичного числа 100.

Представим команды для двоичного числа в десятичной системе:

- Прибавить 1 так и остается +1;
- Добавить справа 0 в десятичной системе означает *2;
- Добавить справа 1 в десятичной системе означает *2+1;

Двоичное число 100 в десятичной системе равно 4;

Двоичное число 11101 в десятичной системе равно 29;

Таким образом условие данной задачи можно записать для десятичной системы и решить аналогичную предыдущей задачу.

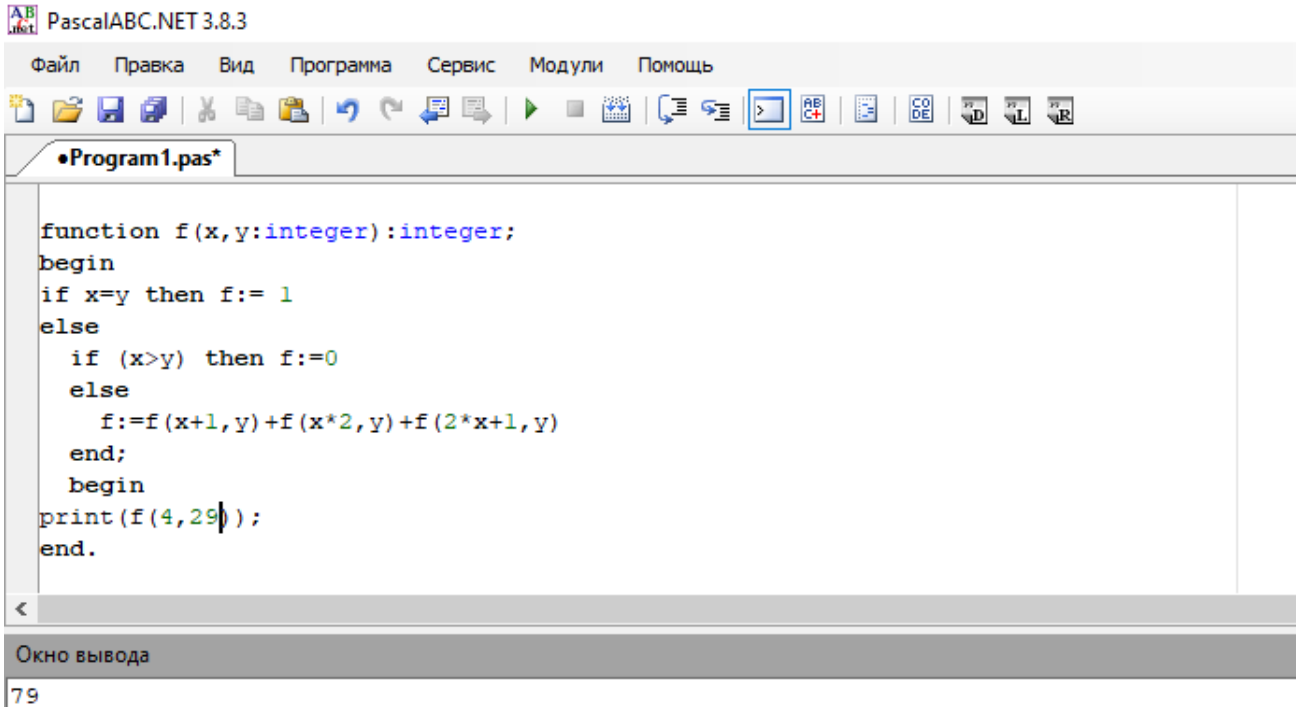


Рис.14. Программный код решения.

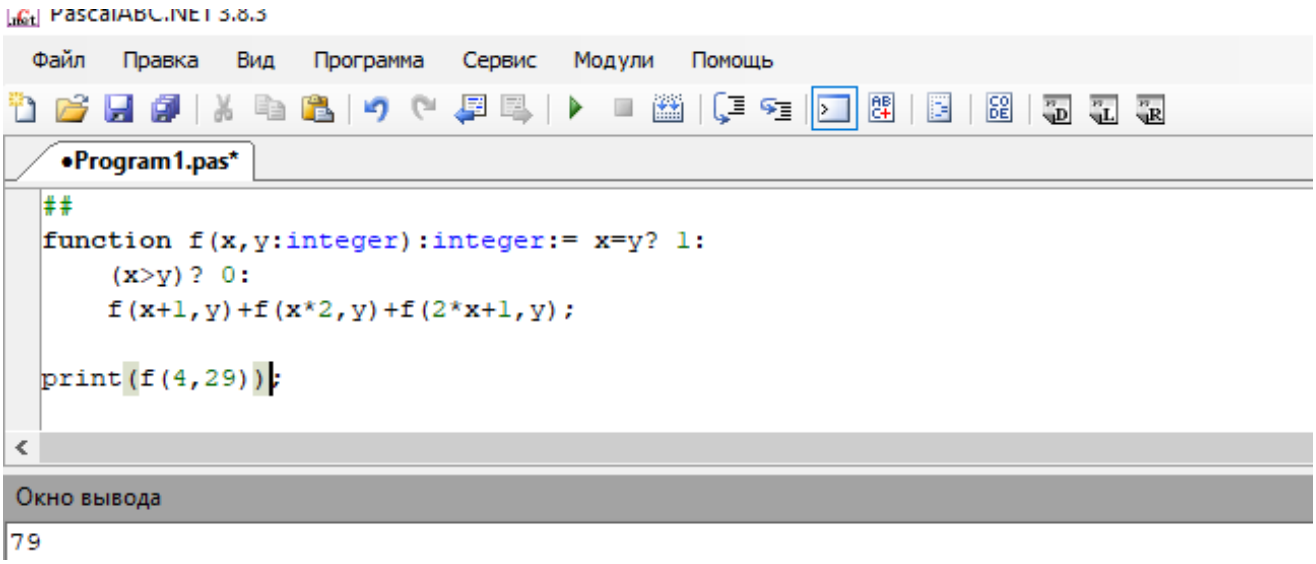


Рис. 15 Краткая запись кода.

Пример 3

Исполнитель Калькулятор преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Прибавить 5
3. Умножить на 3

Определите число, для получения которого из числа 1 существует 175 программ.

У нас есть три команды, по которым из числа 1 можно идти только по возрастанию, т.е. существуют программы при x<y. Описываем функцию и на некотором интервале подсчитываем количество программ. Проверяем условие равенства 175 и выводим результат.

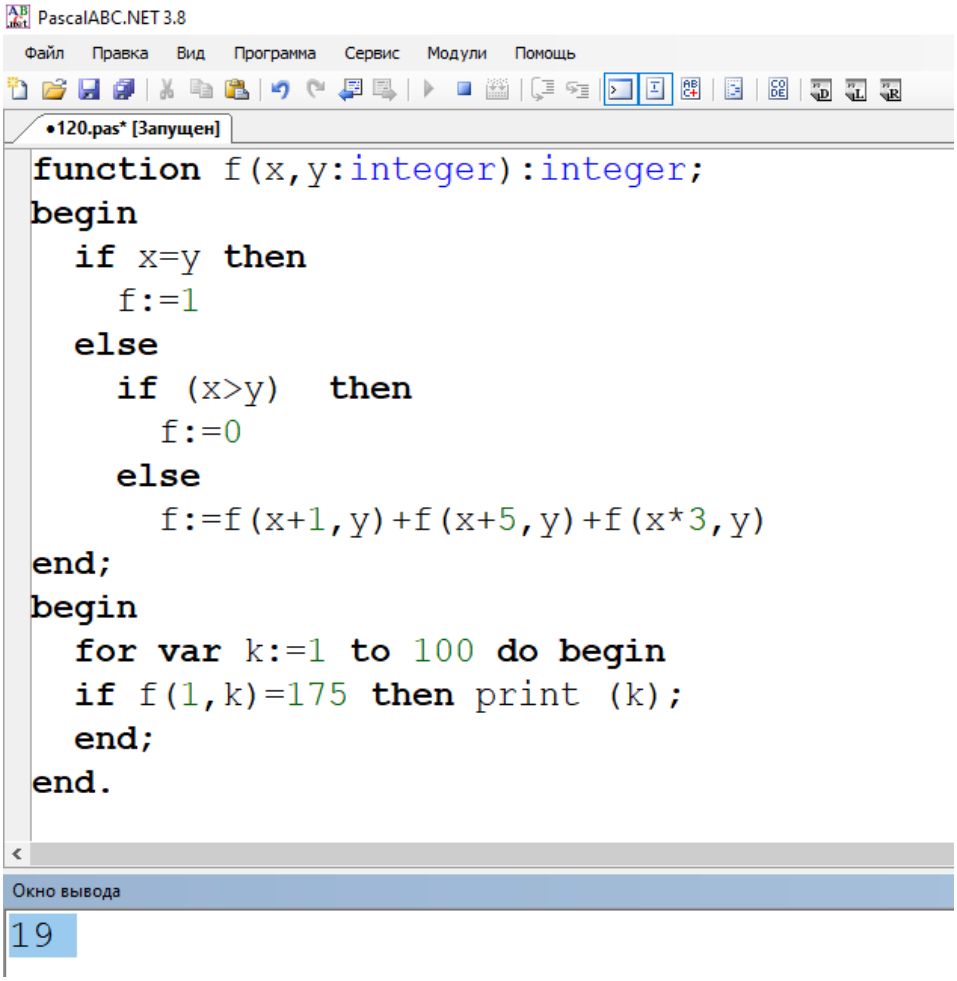
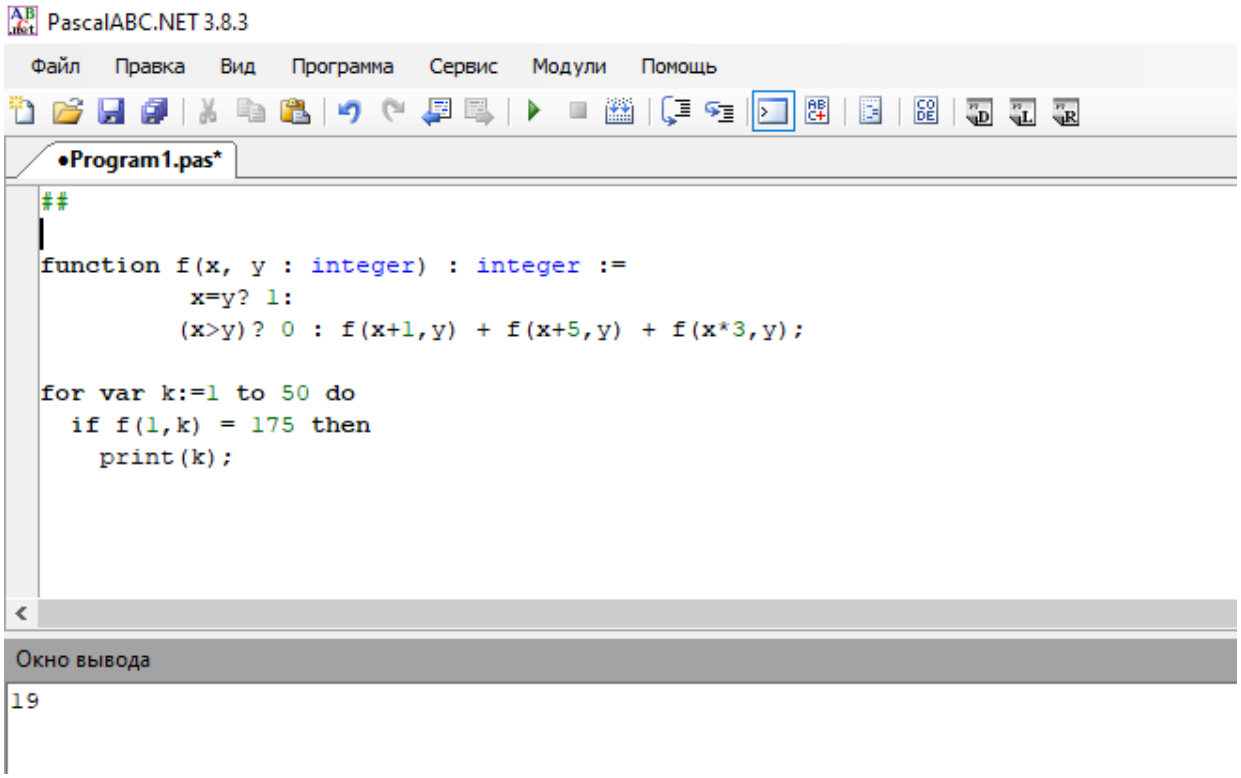


Рис.16 Программный код решения

или так:



Пример 4

У исполнителя Калькулятор две команды, которым присвоены номера:

1. прибавь 1

2. увеличь число десятков на 1

Например: при помощи команды 2 число 23 преобразуется в 33. Если перед выполнением команды 2 вторая с конца цифра равна 9, она не изменяется. Сколько есть программ, которые число 10 преобразуют в число 33?

Если внимательно почитать условие, то очевидно, что на исследуемом отрезке вторая с конца цифра не будет равна 9 и можно проигнорировать это условие, написав стандартный код.

##

function F(a, b: integer): integer := a = b ? 1 : a > b ? 0 : F(a + 1, b) + F(a+10, b);

F(10, 33).Print

Ответ: 25

Если же интервал исследования будет содержать в разряде десятков цифру 9, то этим условием игнорировать нельзя.

Вторая с конца цифра может быть получена делением на 10 остатка от деления числа на 100.

##

```
function F(a, b: integer): integer;
begin
  if a = b then  F:= 1
  else if a > b then F := 0
  else
  begin
    var d := a mod 100 div 10;
    F := F(a + 1, b) + F(if d = 9 then a else 10 * (d + 1) + a mod 10, b)
  end
end;
F(10, 33).Print
```

Ответ: 25

Пример 5

Исполнитель Нолик преобразует двоичное число, записанное на экране. У исполнителя есть две команды, которым присвоены номера:

1. Вычесть 1
2. Убрать последнюю цифру справа

Первая команда уменьшает число на 1. Вторая команда убирает последнюю справа цифру, например, для числа 110 результатом работы данной команды будет являться число 11.

Сколько существует программ, которые исходное двоичное число 110111 преобразуют в двоичное число 110?

Если работать в десятичной системе, то

$110111_2=55$;

$110_2=6$;

Поэтому исполнителю требуется получить из числа 55 число 6.

Вторая команда «Убрать последнюю двоичную цифру справа» в двоичной системе означает в десятичной разделить нацело на 2 .

В функции F нужно условие a > b заменить на a < b, поскольку мы идем от большего значения к меньшему.

##

```
function F(a, b: integer): integer := a = b ? 1 : a < b ? 0 : F(a - 1, b) + F(a div 2, b);
F(55, 6).Print
```

Ответ: 343.

Пример 6

Исполнитель Июнь17 преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1
2. Сделай нечётное

Выполняя первую команду, исполнитель увеличивает число на 1, а выполняя вторую – из числа x получает число 2x+1. Сколько существует программ, для которых при исходном числе 1 результатом является число 25 и при этом траектория вычислений не содержит число 21?

Случай a > b или случай a = 21 обращает нашу функцию в ноль.

##

```
function F(a, b: integer): integer := a = b ? 1 : (a > b) or (a = 21) ? 0 : F(a + 1, b) +
F(2 * a + 1, b);
F(1, 25).Print
```

Ответ: 20.

Пример 7

Исполнитель Июнь17 преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1
2. Сделай нечётное

Выполняя первую команду, исполнитель увеличивает число на 1, а выполняя вторую – из числа x получает число 2x+1. Сколько существует программ, для которых при исходном числе 1 результатом является число 25 и при этом траектория вычислений содержит число 21?

Это пример того, как отсутствие частицы русского языка в условии задачи меняет код и, конечно, ответ.

В этой задаче требуется обязательно проходить через значение 21, мы напишем F(1, 21) * F(21, 25):

```
##
function F(a, b: integer): integer := a = b ? 1 : a > b ? 0 : F(a + 1, b) + F(2 * a + 1, b);
(F(1, 21) * F(21, 25)).Print
```

Ответ: 38.

Пример 8

Исполнитель Калькулятор преобразует число, записанное на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавь 3
2. Умножь на 3

Сколько различных чётных чисел, меньших 100, может получить Калькулятор из исходного числа 3?

Здесь будем перебирать числа в диапазоне от 4 до 98, с шагом 2, т.к. нужны четные и считать подходящие.

```
##
function F(a, b: integer): integer := a = b ? 1 : a > b ? 0 : F(a + 3, b) + F(3 * a, b);
var k := 0;
foreach var b in Range(4, 98, 2) do
  if F(3, b) > 0 then Inc(k);
k.Print
```

Ответ: 16.

Пример 9

Исполнитель Калькулятор преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Умножить на 2
3. Умножить на 3

Программа для исполнителя Калькулятор – это последовательность команд. Сколько существует программ, для которых при исходном числе 1 результатом является число 18?

В рекурсивной функции три ветки:

```
function F(a, b: integer): integer;
begin
  if a = b then Result := 1                    // пришли в B
  else if a > b then Result := 0                 // перескочили B
  else Result := F(a+1, b) + F(a*2, b) + F(a*3, b)   // не дошли до B
end;
```

или:

```
##
function F(a, b: integer): integer := a = b ? 1 : a > b ? 0 : F(a+1, b) + F(a*2, b) +
F(a*3, b);
F(1, 18).Print
```

Ответ 96.

Если задание требует прийти от большего значения к меньшему, вместо отношения a > b указывается a < b.

Пример 10

Исполнитель Калькулятор преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Умножить на 2
3. Умножить на 4

Программа для исполнителя Калькулятор – это последовательность команд. Сколько существует программ, для которых при исходном числе 1 результатом является число 17?

##

```
function F(a, b: integer): integer :=
   a = b ? 1 : a > b ? 0 : F(a + 1, b) + F(a * 2, b) + F(a * 4, b);

F(1, 17).Print
```

Ответ: 54.

Задания для тренировки к №23

1) У исполнителя Калькулятор три команды, которым присвоены номера:

1. прибавь 2

2. прибавь 4

3. прибавь 5

Программа для исполнителя – это последовательность команд. Сколько существует программ, которые число 31 преобразуют в число 51?

2) У исполнителя Калькулятор три команды, которым присвоены номера:

1. прибавь 1

2. прибавь 2

3. прибавь предыдущее

Первая команда увеличивает число на экране на 1, вторая увеличивает это число на 2, третья прибавляет к числу на экране число, меньшее на 1 (к числу 3 прибавляется 2, к числу 11 прибавляется 10 и т. д.). Программа для исполнителя – это последовательность команд. Сколько существует программ, которые число 2 преобразуют в число 9?

3) Исполнитель Июнь15 преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1

2. Прибавить 2

Программа для исполнителя Июнь15 – это последовательность команд. Сколько существует программ, для которых при исходном числе 5 результатом является число 15 и при этом траектория вычислений содержит число 10?

4) Исполнитель Июнь15 преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1

2. Прибавить 3

Программа для исполнителя Июнь15 – это последовательность команд. Сколько существует программ, для которых при исходном числе 3 результатом является число 20 и при этом траектория вычислений содержит число 12?

5) Исполнитель Июнь15 преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1
2. Прибавить 2
3. Прибавить 3

Программа для исполнителя Июнь15 – это последовательность команд. Сколько существует программ, для которых при исходном числе 4 результатом является число 15 и при этом траектория вычислений содержит число 8?

6) Исполнитель Июнь15 преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Умножить на 2
3. Прибавить 3

Программа для исполнителя Июнь15 – это последовательность команд. Сколько существует программ, для которых при исходном числе 4 результатом является число 20 и при этом траектория вычислений содержит число 10?

7) Исполнитель Июнь15 преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Умножить на 2
3. Умножить на 3

Программа для исполнителя Июнь15 – это последовательность команд. Сколько существует программ, для которых при исходном числе 2

результатом является число 28 и при этом траектория вычислений содержит число 7?

8) Исполнитель A13S преобразует целое число, записанное на экране. У исполнителя три команды, каждой команде присвоен номер:

1. Прибавь 1
2. Прибавь 3
3. Прибавь предыдущее

Первая команда увеличивает число на экране на 1, вторая увеличивает это число на 3, третья прибавляет к числу на экране число, меньшее на 1 (к числу 3 прибавляется 2, к числу 11 прибавляется 10 и т. д.). Программа для исполнителя A13S – это последовательность команд.

Сколько существует программ, которые число 2 преобразуют в число 10?

9) Исполнитель Июнь15 преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1
2. Умножить на 2

Первая команда увеличивает число на экране на 1, вторая умножает его на 2. Программа для исполнителя Июнь15 – это последовательность команд. Сколько существует программ, для которых при исходном числе 3 результатом является число 55 и при этом траектория вычислений содержит число 18 и не содержит число 12?

10) Исполнитель Июнь15 преобразует число на экране. У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1
2. Умножить на 2

Первая команда увеличивает число на экране на 1, вторая умножает его на 2. Программа для исполнителя Июнь15 – это последовательность команд. Сколько существует программ, для которых при исходном числе 5 результатом является число 60 и при этом траектория вычислений содержит число 8 и не содержит число 22?

Ответы к заданиям №23

1) 201

2) 57

3) 64

4) 247

5) 308

6) 232

7) 52

8) 39

9) 88

10) 38

Заключение

Данные методические рекомендации по изучению алгоритмических структур, содержащих подпрограммы, циклы и ветвления разработаны на примерах заданий КЕГЭ, предназначены для учащихся 10-11 классов. В них представлена теория, систематизированная по видам алгоритмических структур, рассмотрены задания и способы их решения. Такая структуризация материала позволяет лучше усвоить эту нелегкую для учащихся тему. Разбор типовых задач, примеры их решений, самостоятельные работы с ответами всё направлено не только на преодоление трудностей в изучении темы, но и на организацию качественной подготовки к урокам и к итоговой аттестации.

Данные рекомендации неоднократно отработаны автором на собственных занятиях, этот педагогический момент показал свою эффективность и поможет учителям при подготовке и проведении уроков по теме «Алгоритмические структуры, содержащие подпрограммы, циклы и ветвления», ученикам при подготовке к итоговой аттестации в форме КЕГЭ. Приведенные материалы могут быть использованы как при работе в классе, так и в качестве дополнительных занятий, а также для самостоятельного изучения темы школьниками.

Список использованной литературы

1. Евич Л. Н.. Информатика и ИКТ. Подготовка к ЕГЭ 2021. 10 тренировочных вариантов по демоверсии ЕГЭ 2021 года. Учебное пособие. - Ростов н/Д: Легион, 2021
2. Лещинер В. Р., Крылов С. С. Информатика. Единый Государственный Экзамен. Готовимся к итоговой аттестации. - М.: Интеллект центр, 2021
3. Осипов А. В., PascalABC.NET: выбор школьника. Часть 1, 2, 3. – Ростов-на-Дону, 2020
4. Официальный сайт К. Ю. Полякова https://kpolyakov.spb.ru/school/ege.htm